Path Planning between Two Points for a Robot Experiencing Localization Error in Known and Unknown Environments

Howie Choset, Marco La Civita, Jong Chul Park

Abstract

We use an approach to simultaneous localization and mapping to determine a path between two points for a mobile robot experiencing localization error in a known environment. Simply instructing a robot to move to a point $(x, y)^T$ is not sufficient because the mobile robot's accrued dead-reckoning error prevents the robot from ascertaining its current location and thus it cannot position itself at any location, including the goal. If we knew how dead-reckoning error grew, then perhaps we can direct the mobile robot to a goal location, module an "error margin." Instead, we do not assume an explicit error model, and send the robot to the goal via a sequence of way points, called **meet points**, *whose locations are known a priori. Meet points are nodes of the generalized Voronoi graph and have the property that a robot can use its range sensors to reliably converge on them via a stable control law. The challenge is then to navigate from the final meet point to the goal location (not necessarily on the generalized Voronoi diagram) which constitutes the contribution of this paper. Experiments on the Nomad 200 and soon on the Personal Satellite Assistant Test-bed validate this approach.*

1 Introduction

The three major motion planning problems are point-to-point, mapping, and coverage. Point-to-point determines a path between two prescribed locations; mapping determines a geometric structure which a robot can use to determine a path between two points; and coverage determines a path that directs the robot to pass over all points in a target region. This paper addresses the issue of point-to-point path planning in known (and then unknown) environments for mobile robots that have significant dead-reckoning error.

The motivating application for this work is directing the Personal Satellite Assistant (PSA) which is a free flying robot that will fly inside of the space station. The intended use for PSA will be light material transport and remote inspection for inside the space station. PSA will also be used to help record the actions of astronauts when they are working on various projects inside the space station. PSA will not have a GPS-like positioning system on board and thus will have to use its obstacle sensors to determine its location. Currently, NASA Ames is developing a planar version of PSA that rides on an air-bearing table with obstacles of known location. The NASA Ames researchers will demonstrate PSA driving to designated goal locations using its obstacle sensors to re-localize its position on a known map.

As a start-up problem, we have implemented this approach on a Nomad 200 mobile robot which has some positioning capability, but like all mobile robots, it is fraught with error. This is a commonly studied problem in the mobile robot literature. The approach in this paper uses prior work in simultaneous localization and mapping [3] using generalized Voronoi graphs [4] to direct the robot via a sequence of way points, the meet points (nodes) of the graph, to the goal location.

First, the generalized Voronoi graph is constructed in a simulated model that accurately represents the robot's environment. Next, the meet points along the path are extracted from the graph's path and passed to the robot. In actuality, the distance and direction of the closest objects that define each meet point is passed to the robot. Then, subject to well-defined control laws, the robot uses its sensors to drive from one meet point to the next and upon arriving at each meet point, the robot hones onto the meet point using sensor information. After honing, the robot can zero-out its accrued dead-reckoning error because it knows the location of the meet points a priori from the simulated model.

The robot then sequences through all of the meet points until it reaches the final meet point. From here, the robot traverse an edge of the generalized Voronoi graph until it reaches the departure point, at which point it drives to the goal. The robot uses a control law to hone on the departure point and the goal location. Following the graph's edge merely ensures that the robot will start in a neighborhood of the departure point and goal before honing.

2 Generalized Voronoi Graph: Exploring a CAD Model

Our approach uses the *generalized Voronoi graph* (GVG), a one-dimensional set of curves that captures the salient geometry of the robot's environment. Just as people use roadway systems, the robot uses the GVG to plan a path from a start to a goal, by first planning a path from the start to the GVG, then along the GVG to the vicinity of the goal, and then from the GVG to the goal.

The GVG lends itself nicely to sensor based implementation because it is defined in terms of a distance

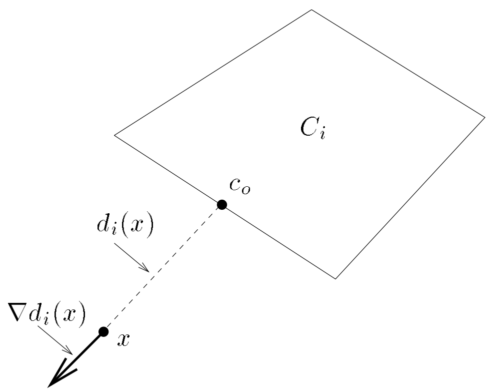

Fig. 1. Distance between x and C_i is the distance to the closest point on C_i. The gradient is a unit vector pointing away from the nearest point.

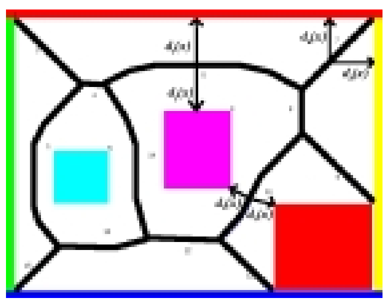

Fig. 2. The solid curve segments are the edges of GVG.

function, $d_i(x) = \min_{c \in C_i} \|x - c\|$, which measures the distance to closest point on object C_i. Simple sonar sensors and laser range finders can provide distance information. In the planar case, GVG edges are simply the set of points equidistant to two obstacles, i.e. the set of points x where $d_i(x) = d_j(x)$. See Figure 2 for an example of planar GVG.

Effectively, if the robot knows the GVG, then it knows the environment. Likewise, if the robot can construct the GVG using sensor data as it moves through the environment, then it has in essence explored the space. The robot uses an adaptation of well-proven numerical curve tracing techniques to generate the GVG. Practically speaking, these techniques trace the roots of the expression

$$G(x) = \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \\ \vdots \\ d_1 - d_m \end{bmatrix} (x) = 0.$$

where d_i is distance to an object C_i, and thus if $(d_1 - d_2)(x) = (d_1 - d_3)(x) = \cdots = (d_1 - d_m)(x) = 0$, the robot is equidistant to m obstacles in an m-dimensional space. In the planar case $G(x) = (d_1 - d_2)(x)$, which is zero when the robot is equidistant to two obstacles. Since G is a function of distance, it can be computed from sensors.

In the plane, the robot generates a GVG edge until it encounters a *meet point* or a *boundary point.* A meet point, as its name suggests, is a node of the GVG where multiple GVG edges terminate (and hence meet). The boundary points are nodes where the GVG edge terminates on the boundary of the environment. When the robot encounters a meet point, it branches its search and generates a new edge until encounters another node. If this node is a meet point, the robot branches its search again and constructs another edge. If the node is a boundary point, then the robot backtracks to the previously visited meet point that has no unexplored edges associated with it and continues edge tracing from there. When all meet points have no unexplored edges, then exploration is *complete.* The underlying structure of the GVG guarantees that the GVG construction procedure will exhaustively explore the entire region.

3 Point-to-Point in Known Static Environment

3.1 Simulator Generates a Path

Initially, the robot uses gradient ascent of distance to the closest obstacles to access the GVG. In other words, the robot moves away from the closest obstacle until it is equidistant to two obstacles. This point is termed the access point. At this point the robot has a choice of two directions to move along the GVG: it chooses the direction that locally decreases its distance to the goal. The robot then searches for the next meet point, where it then chooses among the out-going edges of the meet point that locally decreases the robot's distance to the goal. This procedure is repeated until the distance to the goal is less than the distance to any of the obstacles. This point is termed the departure point. From the departure point, the robot moves in a straight line towards the goal.

The access point, the intermediate meet points, and the departure point serve as the way points through which the mobile robot will pass to achieve a goal location. In addition to outputting way point locations, the simulator also outputs the distance and direction to the three closest obstacles. This information will be used for robot honing on the departure and goal nodes later on. Finally, the simulator outputs a sequence of heading vectors for each meet point. This list of information is then passed onto the robot. See Figure 3

3.2 Accessing the GVG

Incremental accessibility is simply gradient ascent applied to the distance to the nearest obstacle. Since the nearest obstacle is associated with the sensor reading with the smallest value, simply moving in a direction

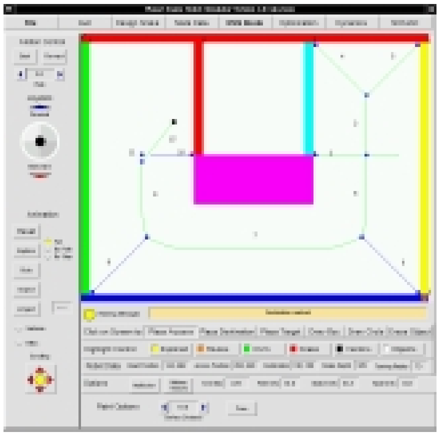

Fig. 3. Simulator model of lab.

opposite to which this sensor is facing results in gradient ascent.

Since the PSA and Nomad robot have a predominant forward direction, the robot must first rotate such that forward direction is pointing away from the nearest obstacle. For robots that have a "small" number of sensors surrounding its perimeter, a simple lookup table can be used to determine the amount of rotating that is necessary. Each entry in the lookup table is indexed by a sensor id. The value of each entry is the amount of rotation necessary to aim the forward direction away from the nearest obstacle.

The robot then drives until the robot is two-way equidistant to two different obstacles. It is not sufficient to sample the two smallest sensor readings to determine the distance to the two closest obstacles because multiple sensors may detect the same obstacle. The minimum distance to each of the obstacles can be approximated by the local minima in the circular array sensor readings. An example is depicted in Fig. 4 where a robot with eight sensors and their measurements are drawn. Sensor H has the smallest value, 10, and is thus pointing at the nearest obstacle. Sensor C is associated with the second closest obstacle because its value is the second smallest local minimum in the sensor array. Note, Sensor A should not be associated with the second closest obstacle because it detects the same object as Sensor H. Nevertheless, the value of Sensor A is not a local minima and thus should not be considered.

In effect, the robot moves until its two smallest local minima are equal, at which point the robot has accessed the GVG. The above claim that the distance to obstacles is the local minima of the sensor array is proven in [1].

3.3 Tracing a GVG Edge

Instead of using numerical step-correct techniques, the robot uses a control law to incrementally construct the GVG. The control law produces smooth paths whereas the numerical methods' result is jagged. In

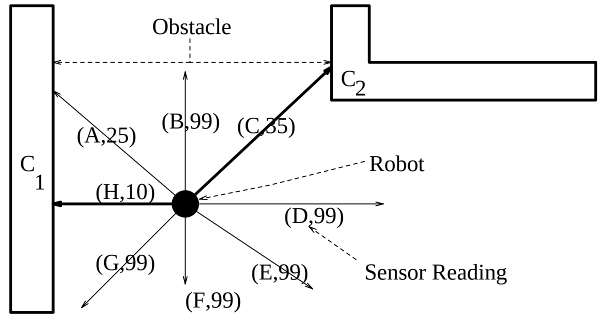

Fig. 4. Sensor Readings

essence, the control law merges the prediction and correction phases. At a point x in the neighborhood of the interior of a GVG edge, the robot steps in the direction

$$\dot{x} = \alpha \, \mathrm{Null}(\nabla G(x)) + \beta (\nabla G(x))^{\dagger} G(x), \qquad (1)$$

where

- α and β are scalar gains,
- $\mathrm{Null}(\nabla G(x))$ is the null space of $\nabla G(x)$,
- $(\nabla G(x))^{\dagger}$ is the Penrose pseudo inverse of $\nabla G(x)$, i.e.,

$$(\nabla G(x))^{\dagger} = (\nabla G(x))^T (\nabla G(x) (\nabla G(x))^T)^{-1}.$$

Note that when x is on the GVG, $G(x) = 0$ and thus $\dot{x} = \alpha \, \mathrm{Null}(\nabla G(x))$. This control law was shown to be stable [2] in some neighborhood of the GVG when $\frac{|\beta|}{|\alpha|} > 1$.

For the mobile robot, the control law to follow the planar GVG (i.e, maintain double equidistance) reduces to

$$\dot{x} = \begin{cases} \alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp}, & \text{if } |d_1(x) - d_2(x)| < \epsilon \\ \alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp} & \\ \quad + \beta(\nabla d_1(x) - \nabla d_2(x))^{\dagger} & \\ \quad\quad (d_1(x) - d_2(x)), & \text{otherwise.} \end{cases}$$

In implementation, the $\alpha(\nabla d_1(x) - \nabla d_2(x))^{\perp}$ corresponds to passing a line through the two closest points on the two closest obstacles, and then taking the line orthogonal to it. Let this direction be v. When the robot is not on the GVG, or not even "close," then $|d_1(x) - d_2(x)|$ exceeds a threshold in which case $\beta(\nabla d_1(x) - \nabla d_2(x))^{\dagger}(d_1(x) - d_2(x))$ participates in determining the heading of the robot. This vector is orthogonal to v and is denoted $v^{\perp}$. The robot then steps $\alpha v + \beta v^{\perp}$ where α and β are the control gains that can be determined empirically. See Figure 5.

3.4 Locating Meet Points

The robot must accurately locate the meet points to capture an accurate topological model of the environment. The robot does not find the exact location of the meet point during the edge tracing process because it is taking finite steps and thus passes by the meet point.

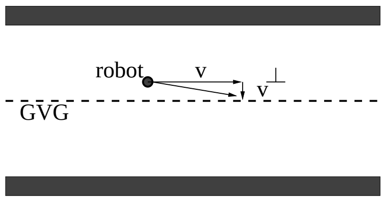

Fig. 5. Control Law for GVG

Also, sensor noise prevents the robot from detecting an exact $m+1$ equidistance location.

Therefore, we have introduced a new meet point honing strategy to bring the robot closer to the exact location of the meet point. This procedure only works when the robot is in the neighborhood of a meet point, which is the case in our implementation. The control law for honing on a meet point is similar to the one for generating new GVG edges, except the G matrix and its Jacobian are

$$G(x) = \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) - d_3(x) \end{bmatrix} \quad \text{and}$$
$$\nabla G(x) = \begin{bmatrix} (\nabla d_1(x) - \nabla d_2(x))^T \\ (\nabla d_1(x) - \nabla d_3(x))^T \end{bmatrix}. \qquad (2)$$

Therefore, $G(x) = 0$ at a meet point, i.e., $d_1(x) = d_2(x) = d_3(x)$. Note that $\text{Null}(\nabla G(x)) = 0$ because, whereas before with the GVG the null space was a one-dimensional line, now the null space is a point. This point corresponds to the origin of the tangent space at x. Therefore, the robot makes the following correction step to hone in on the meet point

$$\dot{x} = \beta \begin{bmatrix} \nabla d_1(x) - \nabla d_2(x) \\ \nabla d_1(x) - \nabla d_3(x) \end{bmatrix}^{\dagger} \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) - d_3(x) \end{bmatrix}$$

which can be shown to be stable using the previous analysis [2].

Geometrically, what is going on is that when the robot is in the vicinity of the meet point, it draws a circle through the three closest points on the three closest obstacles. It then determines the center of that circle and move a differential step towards it. After taking this small step, it repeats this procedure. The stability of the resulting system allows us to conclude that the robot will converge to the location of the actual meet point. See Figure 6

3.5 Locating the Departure Point

The last point the robot must accurately locate before moving towards the goal, is the departure point. This point is the first point on the GVG the robot encounters such that the distance from the goal is less than the distance from any other object in the workspace. The procedure to correctly move the robot onto the departure node has two step: after we leave the last meet point we continue to trace the GVG edge and at the

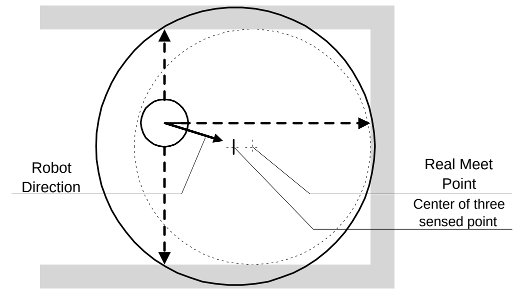

Fig. 6. Meet Point Honing

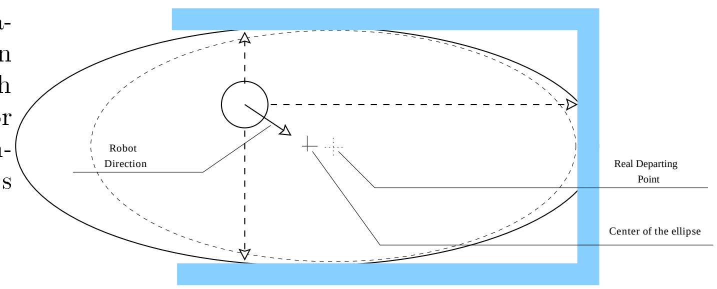

Fig. 7. Departing Point Honing

same time we read the encoders data to calculate the distance of the robot from the last meet point visited; the robot stops when this distance is equal to the known distance between the last meet point and the departure node.

At this point we follow a honing strategy to bring the robot to the departing node. The control law for honing on the departure node is similar to the one for honing on the meet point, except the G matrix and its Jacobian are

$$G(x) = \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) + C - d_3(x) \end{bmatrix} \quad \text{and}$$
$$\nabla G(x) = \begin{bmatrix} (\nabla d_1(x) - \nabla d_2(x))^T \\ (\nabla d_1(x) - \nabla d_3(x))^T \end{bmatrix}. \qquad (3)$$

Therefore, the robot makes the following correction step to hone in on the departure node

$$\dot{x} = \beta \begin{bmatrix} \nabla d_1(x) - \nabla d_2(x) \\ \nabla d_1(x) - \nabla d_3(x) \end{bmatrix}^{\dagger} \begin{bmatrix} d_1(x) - d_2(x) \\ d_1(x) + C - d_3(x) \end{bmatrix}$$

It can be shown that geometrically, what is going on is that the robot, following this control law, moves differential steps toward the center of an ellipse passing for the three closest points, with the direction of the minor axis given by the two closest points and with the major axis passing for the mid point of the segment that connect the two closest points. See Figure 7.

Unfortunately, compared with the meet point control law, there is no simple geometrical way to determine

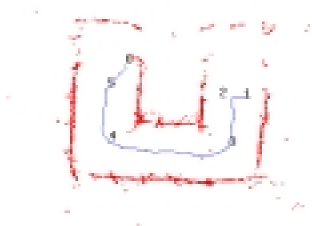

Fig. 8. Mobile robot path for environment depicted in Figure 3.

the center of this ellipse; we need to solve a system of three equations in three unknowns that, alas, is not linear. Due to reasons of speed (all the calculations are performed on line by the robot), we introduce another algorithm to hone in on the departing node. Relying on the fact that at the departing node we know the distances from the three closest objects $d_1(x), d_2(x), d_3(x)$, and the angles between their respective gradient vectors, we can subtract $d_3(x) - d_1(x)$ from the next closest sensor reading, $d_3(x)$, to give the robot the illusion that it is in the vicinity of a meet point.

Once we do that we can use the same meet point honing strategy, to accurately locate the departure node.

3.6 Departing the GVG

Once we have successfully located the departure node, we can zero the error of the encoders and move the robot toward the location of the goal point following a straight line. If the distance of the goal from the departure node is such that we can experience dead-reckoning errors, we can use a honing strategy that is similar to the strategy to hone in on the departure node. The only difference is that this time we have to fake the readings of two sensors: the sensors pointing toward the third and the second closest object. Doing this we are creating two virtual obstacles for the robot that are indiscernible from the real ones. This procedure transforms the goal point in a fake meet point.

See Figure 8 for the resulting mobile robot path.

4 Future Work

4.1 Honing Issues

We have experienced some problems in the procedures to hone in on the departure node and the goal point. These problems are related to the need of measuring the distance to the third closest object in the workspace.

Recall that the distances from objects are calculated looking at the local minima of the sensors ring on the robot. Now this method fails if we are for example in a narrow long corridor where the departing node is in the middle of this corridor. In this case the third object can be so far away from the robot, that there can be false detections of the third local minimum. Suppose that the sensor number n is pointing at the third closest object that is a far wall in our long corridor; to have a local minimum reading at this sensor we should have greater values measured at sensors $n-1$ and $n+1$. But, due to our distance from the third object, the sensors $n-1$ and $n+1$ can actually measure the distances from points on the the lateral walls of the corridor and not the distances from other points on the third object. If this is the case, since the lateral walls are the first and the second closest object, there will be no local minimum in the reading of sensor n, and we fail in identifying the third closest object.

The same problem can appear when we need to hone in on the goal point and the third closest object is located too far from the goal point.

To solve this problem we adopt a simple strategy: since we know the environment, i.e. we know the positions of each object both at the departure node and the goal point, we decided not to use the local minima method to measure the distance from the third object when we are honing on the previous points. What we do is to read the information provided only by the sensor pointing at the third closest object (in the preceding example sensor n).

This procedure worked in the demo run in our lab, but again has some kind of limitations. In particular if we are traversing a long corridor (longer than the one that gives us problem with the local minima method) we can not have correct measure distances even from the sensor pointing directly at the third closest object. This is due to the fact that a sonar sensor measures distances from objects that are in a well defined cone coming out from the sensor itself. In the case of Polaroid sonar sensors that are mounted on the Nomad 200 the magnitude of the cone's angle is 22.5 degrees. So it may be that this cone intersects the lateral walls before intersecting the third closest object, giving as a result a wrong distance measure.

4.2 Weak Meet Points

To its advantage, the algorithm uses the environment to define its landmarks. Unfortunately, this dynamically discovered landmark approach has a weakness: for some environments, a slight change to the environment can result in a radically different graph representation of the environment. This problem manifests itself when errors in sonar sensors cause the robot to oscillate between different environment representations. In other words, the environment remains fixed, but a slight change in sensor readings give the robot the illusion that the environment slightly changed. For the GVG, the problematic landmarks are called *weak meet points*, those that are sometimes sensed and sometimes ignored.

An "unscheduled" meet point may correspond to an unmodeled object moving around a known location. The existence of weak meet points can confuse the robot into thinking a dynamic obstacle is present. In static environments, we conjecture that weak meet points are not a problem because we can identify them (within some error box) and then move onto the next meet point, at which time we can eliminate error. We assume this in our current implementation.

5 Conclusion

The generalized Voronoi graph (GVG) is a roadmap of a robot's environment that the robot can use to plan a path between two points in three steps: (1) determining a path onto the GVG, (2) finding a path along the GVG, and then (3) charting a path from the GVG to the goal. Originally, the GVG was used for sensor based exploration of unknown spaces because if the robot can generate the GVG using sensor data, it can then use the GVG to plan paths in a previously unexplored environment.

A feature of the GVG is that it has geometries encoded in it that help the robot localize itself while exploring unknown regions. These features are called *meet points* which are nodes of the GVG. While exploring an unknown space, each time the robot encounters a meet point, the robot stores a sensor signature of the meet point that the robot can use when the robot reencounters the meet point. In many scenarios, many meet points "look the same," so the robot has to use the adjacency relationships among the meet points to determine its location in the partially explored GVG. In other words, if the robot is at meet point 11, which looks like meet point 1, and meet point 12, which looks like meet point 2, are adjacent to each other, *and* meet points 1 and 2 are adjacent, then the likelihood that 11 is 1, and 12 is 2 increases. Essentially, we are doing graph matching to achieve localization.

This paper used the GVG simultaneous mapping and localization to address the problem of path planning between two points for a mobile robot experiencing localization error. Instead of using the meet points to do graph matching, the meet points serve as landmarks for the robot to eliminate its accrued localization error. First, a path is generated using the GVG in a known environment. A linked list of meet points, each with a sensor signature and pointer to the next meet point, is passed to the robot. The robot uses a control law to follow the edge connecting adjacent meet points and then a control law to hone in on each meet point. Once the honing process is complete, the robot can look up the true location of the meet point and zero-out the localization error that has accrued since visiting the previous meet point.

The control laws all use distance information to nearby obstacles. Sometimes, limitations in range and azimuth resolution prevent the robot for measuring this distance, so more complicated sensor processing is invoked. Already, we have developed a method to improve the resolution of sonar sensors, but using a laser ranger would immediately solve this problem.

Another problem with this method deals with "weak" meet points; these are nodes that can appear or disappear seemingly at random because of slight fluctuations in sensor readings. Although these meet points are rare, they can pose a serious problem when they occur because the robot navigates using meet points. This is a current area of research as well, and we feel that this is not a property of the GVG method, but that of all graph-based localization techniques.

Future work will include implementing this algorithm on the actual Personal Satellite Assistant planar prototype at NASA Ames. After that, we will extend the result in this paper to three-dimensional path planning for a mobile robot experiencing localization error.

Acknowledgements

The authors gratefully acknowledge Keiji Nagatani at Okayama University for his impact on this project and the software he developed for it. Also, John Loch at NASA Ames provided the authors with some insight and inspiration that resulted in this paper. Finally, conversations with Mark Sibenac and Hans Thomas at NASA Ames were invaluable for this paper.

References

[1] H. Choset, I. Konuksven, and J.W. Burdick. Sensor Based Planning for a Planar Rod Robot. In *Proc. IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion on Multisensor Fusion and Integration for Intelligent Systems*, Washington, DC, 1996.

[2] H Choset, I Konuksven, and A Rizzi. Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph. In *Proc. of IEEE Int. Conf. on Autonomous Robots*, Monterey, CA, 1997.

[3] J. J. Leonard and H.F. Durrant-Whyte. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, 1991, May 1991.

[4] K. Nagatani, H. Choset, and S. Thrun. Towards Exact Localization without Explicit Localization with the Generalized Voronoi Graph. In *IEEE Int. Conf. on Robotics and Automation*, Lueven, Belgium, May 1998.